Vokasi Unesa Bulletin of Engineering, Technology and Applied Science (VUBETA)
https://journal.unesa.ac.id/index.php/vubeta
Vol. 2, No. 1, 2025, pp. 67~76 DOI: 10.26740/vubeta.v2i1.34153
ISSN: 3064-0768

`

Exploring Supervised Learning Methods for Predicting Cuisines from Their Ingredients

Yonathan Ferry Hendrawan[1*], Omkar Chekuri[2]
[1]Informatics Department, Engineering Faculty, Trunojoyo University, Bangkalan, Indonesia
[2]School of Computer Science, Gallogly College of Engineering, University of Oklahoma, Norman, United States of America

Article Info

Article history:

Received November 27, 2024
Revised January 10, 2025
Accepted February 15, 2025

Keywords:

Classification
Cuisine Prediction
Supervised Learning
Methods Comparison
Support Vector Machine

ABSTRACT

This study explores the use of multi-class classification to predict cuisines based on ingredient list using a Kaggle dataset derived from the Yummly recipe database. The goal was to identify the most effective machine-learning techniques for classifying recipes into different cuisine regions based on their ingredients. Six supervised learning methods were examined: Backpropagation Neural Network, Support Vector Machine (SVM), Naive Bayes, Decision Tree, Random Forest, and AdaBoost. The preprocessing pipeline involved tokenizing ingredients into numerical features, ensuring compatibility with machine-learning algorithms, and facilitating model training and evaluation. Among the models tested, the SVM and Random Forest algorithms performed the best, achieving accuracies of 76.7% and 73.2%, respectively. These results were relatively close to the top competition leaderboard accuracy of 83%. Our custom implementations of the Backpropagation Neural Network and Decision Tree demonstrated competitive performance, though hardware limitations during experimentation prevented the full optimization of these models. The findings emphasize the critical role of factors such as parameter tuning, dataset size, and feature preprocessing in determining classification accuracy. Additionally, the study highlights how a combining of well-selected algorithms and data preprocessing can yield meaningful improvements in prediction quality. All codes and materials used in this research are publicly available, enabling further exploration by other researchers and practitioners.

1. INTRODUCTION

AI-assisted cooking is a modern way to enhance your cooking experience. With AI, you can discover new recipes, optimize ingredient usage, and create innovative dishes. Despite the challenges, the benefits offered by this technology are significant, especially for those who want to cook more efficiently and creatively [1]-[6]. Each region has its own unique cuisines. . This is reflected in the ingredients used in cooking; for example, cheese is common in Italian cuisines, and cumin is common in Indian cuisines. Kaggle created a competition regarding this condition [7]-[12]. Its main objective is to build machine learning methods to solve the multi-class classification problem and predict cuisines based on the ingredients. Many solutions have been submitted, and the highest accuracy score from the Kaggle leaderboard website is 0.83216.

In our research project in this paper, we acquired the competition dataset from the Kaggle website. The dataset itself comes from the Yummly website, which provides recipe recommendations. The data uses JSON format with a total size of 2.2 MB. The training data consists of a unique recipe id, type of cuisine, and the list of ingredients that belong to the cuisine. The test data for the type of cuisine field were omitted. The goal is to predict the cuisine type, given the ingredients correctly.

For this research, we explored six supervised learning methods: Backpropagation Neural Network [13][14], Support Vector Machine [15][16], Naive Bayes [17][18], Decision Tree [19][20], Random Forest

*Corresponding Author
Email: yonathan.hendrawan@trunojoyo.ac.id

[21]-[23], and AdaBoost [24]-[26]. We evaluated each method's result by submitting it to the Kaggle competition website and acquired its accuracy and rank.

RecipeDB is a food recipe database developed with the help of Artificial Intelligence (AI) and Machine Learning (ML) technology. This database is designed to provide structured information about various food recipes from around the world, including details of ingredients, cooking methods, nutritional values, and relationships between certain ingredients. RecipeDB aims to help users find recipes that suit their preferences, such as special diets, allergies, or ingredients available in the kitchen [27]-[29]. Sharma et al. built a set of systems to learn ingredients, instructions, and utensils and used them to predict the cuisine's region from the RecipeDB dataset [19]. The methods used are Naive Bayes, Logistic Regression, Support Vector Machine, Random Forest, and Recurrent Neural Network. Our research uses a similar approach where we explored several classification methods to solve the Kaggle's cuisine dataset.

Fausett gives a thorough explanation regarding of Neural Networks [30]. We used this as the primary reference when we built our Backpropagation implementation. Meyer et al. conducted an SVM benchmarking study, comparing its performance to 16 classification and 9 regression methods [31]. Although SVM results were not the best among the three experiments, SVM produced a good overall result, especially for classification. This paper motivated us to use SVM in our research. We wanted to see whether our Neural Network implementation can outperform SVM or not in classifying the cuisines. Naive Bayes is a Machine Learning method whose underlying feature-independent assumption might not work well in many cases. However, Domingos and Pazzani showed that it can perform well even when strong attribute dependencies exist [32]. We wanted to test this on our cuisine problem, since we think the ingredients are not independent of each other, especially in regional cuisines. Patel et al. describes various types of Decision Trees, their applications, libraries, and issues [33]. We used this paper as our entry to understand the landscape of the Decision Tree method.

Breiman explains the advantages of random forests and how the noise in the model will be reduced by using a random set of features instead of using all the features [34]. The author also describes how using many trees will let the random forests converge well and reduce overfitting. This paper helped us understand the mathematical foundations and advantages of Random Forests and the reasons behind the generalizability of the models. Hastie et al. propose a variant of the AdaBoost algorithm called SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function) that directly extends to multi-class classification without reducing it to multiple two-class problems by adjusting the weight term used for boosting the misclassified examples [35]. This algorithm is equivalent to a forward stagewise additive modeling algorithm that minimizes an exponential loss function for multi-class classification. Our AdaBoost implementation uses SAMME for multi-classification.

There are three contributions to this research. The first is the comparison of six supervised learning methods' accuracy in solving the cuisine prediction problem. The second is the exploration of different parameters' effects on the learning methods. The last is producing and publishing relevant code and other materials that are openly available at https://github.com/OmkarChekuri/CuisineSL.

In the following sections, we start by detailing the supervised learning methods explored, including Backpropagation Neural Network, Support Vector Machine, Naive Bayes, Decision Tree, Random Forest, and AdaBoost, emphasizing their implementations and the reasoning behind their selection. Following this, we discuss the preprocessing techniques used to transform the Kaggle dataset for model training and testing. We then present the experimental setup in section 3, including parameter tuning and accuracy evaluation, and a comparative analysis of model performance. This is followed by the discussion section that delves into insights gained from the results and the limitations encountered. Finally, the conclusion summarizes the research contributions and suggests avenues for future work, highlighting the open-source resources we have provided to encourage further exploration.

2. METHOD

Backpropagation is a neural network that computes the gradient of the loss function and minimizes the error between the input and output set. In this research, we used learning rate = 0.1, a single hidden layer with 30 and 60 neurons in it. We recorded the impact of using a different number of neurons on the model's performance.

Support Vector Machine is a supervised learning method that classifies the data into spaces limited by clear gaps as wide as possible. It can perform linear and non-linear classification. In this research, we conducted an experiment where we varied the percentage of training data used and recorded the results.

Naive Bayes is a method to classify data using a probabilistic model. Our experiment used three different types of Naive Bayes: Gaussian, Bernoulli, and Multinomial. We wanted to know which one produced the best result for cuisine data whose features are not independent. We used the ID3 (Iterative Dichotomizer 3) algorithm to build a decision tree. The ID3 algorithm builds the decision tree using a greedy approach. It begins

by assigning all the labels to a root node and calculating the entropy of the root node. It then makes a split based on the attribute that produces the highest information gain by looping through all the attributes. The tree gets built recursively at each node until the leaf nodes are pure.

Random Forest is a type of ensemble machine learning algorithm. Random forests build several decision trees by taking a random sample of the training data with replacement and using a constant number of variables or attributes which are selected at random and building three to the most significant extent possible. For classification, the algorithm takes the majority vote of the decision trees and makes the prediction.

AdaBoost classifier builds a classifier and then adjusts the weights of the examples by increasing the weights of the samples that were classified incorrectly and reducing the weights of the samples sampled incorrectly such that the weak learners focus more on improving the predictions. The word boosting essentially refers to boosting the weights of misclassified training data. We trained different models with various numbers of estimators: [10, 20, 40, 80, 100, 200].

Each model is developed using the sci-kit-learn library so that we can conduct experiments. We also chose to build Backpropagation and Decision Tree from scratch since we had some experience regarding those two methods and wanted to build them personally. Both are library-based, and our implementation is available on the GitHub repository.

3. RESULTS AND DISCUSSION

The data acquired from the Kaggle website went through three significant steps. The first is Preprocessing, in which the data is loaded, cleaned, and formatted for the next step. The second is Learning in which the training data was used to train the models. The last step is to use the model to classify the test data and produce the CSV file for accuracy submission on the Kaggle website.

Preprocessing:

The data is in JSON format with many overlapping ingredient values. We created codes to read the data file and split the cuisine and ingredients pairs into separate arrays. Next, the ingredients must be grouped together to make an array in which each element is unique. Using this array, we convert each ingredient in training data to a numerical value based on its index. For this step, we used the count vectorizer function available in the Sci-kit Learn library to convert the ingredient array into an array of token counts. This array is the one we used to train the models. We used the same vocabulary obtained from the previous tokenization process to make the testing data from the test.json file.

3.1. Results

After preprocessing, the dataset was split into 39,774 training examples with 6,867 features (ingredients) and 9,944 test examples for predictions. Various machine learning models were evaluated with specific configurations, recording both accuracy and runtime for a comprehensive comparison. Below are the results and observations for each model.

Table 1. The hidden layer neuron number and number of iteration experiment result

Hidden Layer Neurons	Iteration	Accuracy	Time (seconds)
30	10	0.61886	413.99
	50	0.71671	1949.11
	100	0.72365	4133.63
	200	0.72718	7937.19
	300	0.72878	12116.08
	400	0.72898	15771.77
60	10	0.62500	709.08
	50	0.71530	3835.36
	100	0.72214	7196.27
	200	0.74004	14512.08
	300	0.74255	20995.17
	400	0.74155	26979.08

3.1.1. Back Propagation

We experimented with varying the number of hidden layer neurons and training iterations, as shown in Table 1. The results indicate that increasing the number of iterations generally improves accuracy, but only up

to a point, after which overfitting occurs. Training with 60 neurons and 300 iterations achieved the highest accuracy of 74.26%, though it required a significant runtime of approximately 5.8 hours. Table 1 summarizes the training time for different configurations and the corresponding accuracy obtained from submissions to the Kaggle website.

From the experiment result, we get that the higher the iteration, the scores generally, the higher, except for 400 iterations with 60 neurons. We suspect that this condition is caused by overfitting our model. We stopped our experiment at 400 iterations because of the overfitting suggestion and the long duration of running time. It took 7.5 hours to complete the last experiment on our system.

3.1.2. Support Vector Machine

Next, we build a Support Vector Machine model. The result is shown in Table 2. We varied the percentage of the training data used. This is because we used 100% of the training data in our first attempt and did not receive any results after several hours. We thought that we made some mistakes in the program. So, we decided to stop the running process and redid the training with much less data: 2.5%. Our code was OK, and the SVM model just needed some time to finish. Then, we tried to increase the training data percentage to 25%. After gaining familiarity with the running time, we rerun it for all training data. It took 4.5 hours for the program to give the result. The result shows that the higher the percentage of training data used, the higher the accuracy.

Table 2. The Support Vector Machine experiment result

% of Training data used	Accuracy (%)
2.5	0.47747
25	0.69489
100	0.76769

3.1.3. Naïve Bayes

For the Naïve Bayes experiment, we built Gaussian, Bernoulli [36], and Multinomial models [37] for the Sci-kit Bayes experiment. The result is shown in Table 3. The Multinomial model achieves the best result. The good part of Naïve Bayes is that it can reach comparable accuracy in a much shorter running time compared to the previous experiments.

Table 3. The Naïve Bayes three types experiment result

Type	Accuracy	Time(seconds)
Gaussian	0.34332	13.05
Bernoulli	0.70917	118.45
Multinomial	0.73722	118.45

3.1.4. Decision Tree Algorithm

In our initial experiments using the Scikit library for decision trees on the current cuisine prediction dataset, we achieved better accuracy with the Gini index for calculating information gain and a max depth of 500 for the tree. We used the same parameters to build our implementation of the Decision tree. We achieved the results shown in Table 4 for the Decision Tree without using the Maximum Depth parameter and Table 5 for the Decision Tree with a Maximum Depth of 500. Due to the large number of features and examples, the Decision tree algorithm took a very long time to train.

Table 4. Decision Tree experiment results without max depth parameter

% of Training data used	Accuracy	Max Depth	Time (seconds)
2.5	0.4243	None	99.2
25	0.5496	None	2820.0
50	0.5896	None	8487.7
100	0.61745	None	21972.8

`

Table 5. Decision Tree Experiment results with max depth 500

% of Training data used	Accuracy	Max Depth	Time (seconds)
2.5	0.43714	500	243.0
25	0.58055	500	9472.5
50	0.60488	500	27356.3
100	0.62745	500	126458.5

We also implemented a Decision tree algorithm in the Scikit learn library using full data Gini index criteria and a maximum depth of 500 to understand how it performs compared to the algorithm we developed. We used different built-in hyperparameters, such as minimum samples in the leaf node to split the tree and minimum samples in the leaf node to improve the model. We found that the performance of the decision tree remained mostly unchanged, producing an accuracy of 60% to 61%, which is very close to our model. Table 6 shows the results of our experiments.

Table 6. Scikit learn Decision tree Experiment results

Min samples at leaf Node	Min samples required to split	Accuracy
3	2	0.60985
5	2	0.61559
10	2	0.61126
2	5	0.60382
5	10	0.61518

3.1.5. Random Forests

We experimented with maximum features for the Random Forests, which equals the square root of the total number of features. This reduced the features used by the model from 6867 to 83 and improved the running time from hours to minutes and seconds in some cases. The performance of Random forests improved by increasing the number of Decision trees called estimators. The result is shown in Table 7. Due to hardware limitations, we could not run models with 200 trees and beyond, which resulted in system failure.

Table 7. Random Forests Experiment results with increasing estimators and constant features

Estimators	Accuracy	Max Features	Time (seconds)
20	0.7026	83	41
50	0.72294	83	101
100	0.72727	83	201
150	0.73189	83	302

We also ran experiments by varying the Max Features while keeping the Estimator number constant and achieved the results in Table 8. The best results are achieved by using a smaller number of Max Features. The time required in Table 8 is higher than that in Table 7 since the number of Max Features is significantly higher.

Table 8. Random Forests Experiment results with increasing features and constant estimators

Estimators	Accuracy	Max Features	Time (seconds)
50	0.69398	1000	357.0
50	0.68694	2000	625.7
50	0.68302	3000	1101.5

3.1.6 Ada Boosting:

For the Ada Boosting experiment, we used the algorithm called AdaBoost-SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function developed by Hastie et al. [35]. We experimented with varying decision stumps and found that the accuracy improved slowly over time at a much slower rate. The result is shown in Table 9.

Table 9. Ada Boosting Experiment results

Number of Estimators	Accuracy
10	0.3265
20	0.4183
40	0.4183
80	0.5370
100	0.5548
200	0.5545

3.2 Discussion

The performance of our built Neural Network is just a little under 75%. We deemed this result good enough, considering that the highest accuracy obtained on the website leaderboard was 83%. The increase of hidden layer neuron number raised the accuracy a little bit. It needs to be noted that our Neural Network implementation uses only one hidden layer. LeCun et al.[38] and Krizhevsky et al.[39] have shown that adding more hidden layers will significantly improve the performance of classification tasks for deep learning methods. We left adding more hidden layers to the future work. We also suspect that the training data size influenced the accuracy, as neural networks tend to perform poorly with small datasets, as shown by Nguyen et al. [40].

The SVM model outperformed the Backpropagation results in our experiments. This result is similar to Chen and Wu's [41]. Nguyen et al.[40], Vapnik [42], and Caruana et al.[43] also, SVM might outperform the neural networks for smaller datasets as they can generalize well with small datasets. Neural networks may struggle with overfitting unless specific techniques like regularization are applied. Furthermore, deep learning methods require large datasets to avoid overfitting, as observed by Chollet [44] , Tang and Russ [45].

Our Naïve Bayes can reach more than 70% accuracy. This result shows that Naïve Bayes can perform well in cuisine data that we suspect to have dependency features. Since Gaussian Naïve Bayes is geared toward continuous values, it did not perform well in our experiment. Also, Gaussian Naïve Bayes assumes that the continuous features follow a normal (Gaussian) distribution. If the data does not meet this assumption, the model's performance can degrade significantly [46]. Applying transformations to make the data more Gaussian-like, such as log transformation or Box-Cox transformation, can sometimes improve GNB performance [47]. Multinomial and Bernoulli performed better since they are discrete data classifiers, fitting for our discrete cuisine ingredient data. The integer features of the dataset are more suitable for Multinomial performance since it uses integer counts than Bernoulli, which is designed for binary features. The fast running of the Naïve Bayes suggests that the algorithm is computationally efficient and it requires less training data. Our results showed that Bernoulli and Multinomial have the same running time, suggesting they have similar time complexity. We believe the difference will become prominent for larger datasets. Manning et al.[37] also suggests the same. Further, Naïve Bayes have shown to work well for high dimensional data such as the text data by Rennie et al. [48] and McCallum et al. [49].

`

Our results show that the Decision Trees's accuracies have more than 10% difference compared to the Neural Network's. We suspect this is because decision trees perform better for lower dimensional data; and with high dimensional data, they tend to overfit. Decision trees are based on the statistical measure of entropy, and thus they do not account for correlation in high dimensional data. Raileanu et al.[50] have argued that this might be due to the fact that decision trees partition the data space using axis aligned splits, which can be insufficient for capturing intricate relationships. A single incorrect node can drastically reduce the performance of the tree. We ran the same set of experiments with limiting the maximum depth of the tree to 500 and without limiting the maximum depth of the tree to see if we can achieve better performance. We achieved slightly better performance with limiting the Maximum depth of decision tree to 500. It is also not surprising that a neural network model works well for natural language processing tasks such as text classification, as argued by Goldberg [51].

For the Random Forests experiment, we achieved an accuracy of 73.1% with just 150 trees with only 83 features, which is better than neural networks in some experiments. Its high accuracy can be attributed to how the decision tree handles noise and overfitting [34]. Random forests are less noise-sensitive by incorporating randomness in features and selecting examples. Random forests achieve higher convergence by increasing the number of decision stumps used in training the data due to the law of large numbers, and they are stable. This method is very fast because it uses only a subset of data and features. Ho [52] and Liaw [53] demonstrated through experiments that random forests handle high dimensional data well compared to single decision trees. However, random forests have low interpretability due to their ensemble nature compared to a single decision tree. Our results also suggest that a neural network with a single layer has outperformed the random forest with 150 trees, suggesting that neural networks have very high expressive power [39]. Random forests are computationally much more efficient compared to neural networks.

We used AdaBoost as our boosting algorithm and Random forests as our Bagging algorithm. . Boosting involves training a classifier with the whole dataset, adjusting the weak classifier's weights to improve the performance, and then retraining the model iteratively. Unlike in a random forest, training can only occur sequentially, where training can be performed parallelly with only a subset of data. Our results showed that the convergence of random forests is very fast compared to that of AdaBoost. Our AdaBoost results also indicate that it could not handle noise and overfitting efficiently with fewer trees. We believe this is due to the highly correlated nature of data and the high dimensionality. Experimental studies by Dietterich [54], Meir and Rätsch [55] have shown that AdaBoost can sometimes overfit, especially if the weak learners are too complex or the data is too noisy. In comparison, random Forest is generally robust[53] and better at avoiding overfitting due to the averaging of multiple trees.

4. CONCLUSION AND LIMITATION

This study investigates the use of machine learning models to predict cuisines based on ingredient lists, evaluating six popular supervised learning methods: Backpropagation Neural Network (BPNN), Support Vector Machine (SVM), Naive Bayes, Decision Tree, Random Forest, and AdaBoost, using a Kaggle dataset. The findings show that SVM and Random Forest performed the best, with accuracies of 76.7% and 73.2%, respectively, suggesting that these models are well-suited for cuisine classification tasks. The research builds upon previous studies by offering a comparative analysis of multiple algorithms and introducing insights into the importance of data preprocessing, parameter tuning, and feature transformation in improving model performance. While BPNN and Decision Trees showed competitive results, hardware constraints limited their optimization. The study's limitations include model generalization and data preprocessing challenges, which may affect performance with larger and more diverse datasets. Future research could focus on improving feature engineering, experimenting with deep learning models, and optimizing hyperparameters for better performance. Further exploration with larger, more varied datasets and real-time prediction systems could provide valuable contributions to the field.

REFERENCES

[1] H. Kim, S. Choi, and H. H. Shin, "Artificial intelligence in the kitchen: can humans be replaced in recipe creation and food production?," *Int. J. Contemp. Hosp. Manag.*, 2025. https://doi.org/10.1108/IJCHM-04-2024-0549.

[2] P. M. A. Sadique and R. V Aswiga, "Automatic summarization of cooking videos using transfer learning and transformer-based models," *Discov. Artif. Intell.*, vol. 5, no. 1, p. 7, 2025. https://doi.org/10.1007/s44163-025-00230-y.

[3] A. D. Starke, J. Dierkes, G. Lied, G. A. B. Kasangu, and C. Trattner, "Supporting healthier food choices through AI-tailored advice: A research agenda," *PEC Innov.*, p. 100372, 2025. https://doi.org/10.1016/j.biombioe.2025.107620.

[4] O. Awogbemi and D. A. Desai, "Application of computational technologies for transesterification of waste cooking oil into biodiesel," *Biomass and Bioenergy*, vol. 194, p. 107620, 2025.

[5] V. Moglia, O. Johnson, G. Cook, M. de Kamps, and L. Smith, "Artificial intelligence methods applied to longitudinal data from electronic health records for prediction of cancer: a scoping review," *BMC Med. Res. Methodol.*, vol. 25,

no. 1, p. 24, 2025. https://doi.org/10.1186/s12874-025-02473-w.
[6] A. Umar, A. Ore-Ofe, I. Ibrahim, A. A. Abiola, and L. A. Olugbenga, "Development of a Head Gesture-Controlled Robot Using an Accelerometer Sensor," *Vokasi Unesa Bull. Eng. Technol. Appl. Sci.*, pp. 93–102, 2024. https://doi.org/10.26740/vubeta.v1i2.35114.
[7] G. O. Chukwurah *et al.*, "Cultural Influence of Local Food Heritage on Sustainable Development," *World*, vol. 6, no. 1, p. 10, 2025. https://doi.org/10.3390/world6010010.
[8] S. Højlund and O. G. Mouritsen, "Sustainable Cuisines and Taste Across Space and Time: Lessons from the Past and Promises for the Future," *Gastronomy*, vol. 3, no. 1, p. 1, 2025. https://doi.org/10.3390/gastronomy3010001.
[9] G. El Majdoubi and H. El Ayadi, "Unlocking Potential: A Holistic Approach to Water, Energy, and Food Security Nexus in Tangier-Tetouan-Al Hoceima, Morocco," *Case Stud. Chem. Environ. Eng.*, p. 101129, 2025. https://doi.org/10.1016/j.cscee.2025.101129.
[10] H. Karg, I. Bellwood-Howard, and N. Ramankutty, "How cities source their food: spatial interactions in West African urban food supply," *Food Secur.*, pp. 1–22, 2025. https://doi.org/10.1007/s12571-025-01518-8.
[11] G. Ares *et al.*, "Food outlets in Montevideo: Implications for retail food environment research in the majority world," *J. Nutr. Educ. Behav.*, 2025. https://doi.org/10.1016/j.jneb.2024.12.011.
[12] Z. Z. A. Thaariq, "Internet of Educational Things (IoET): An Overview," *Vokasi Unesa Bull. Eng. Technol. Appl. Sci.*, vol. 1, no. 2, pp. 81–92, 2024.
[13] X. Ma, H. Ning, X. Xing, and Z. Zang, "Co-pyrolysis behaviors and reaction mechanism analyses of coal slime and moso bamboo based on GC/MS and backpropagation neural network," *Int. J. Hydrogen Energy*, vol. 98, pp. 197–210, 2025. https://doi.org/10.1016/j.ijhydene.2024.12.051.
[14] S. K. Mondal *et al.*, "Glacial lakes outburst susceptibility and risk in the Eastern Himalayas using analytical hierarchy process and backpropagation neural network models," *Geomatics, Nat. Hazards Risk*, vol. 16, no. 1, p. 2449134, 2025. https://doi.org/10.1080/19475705.2024.2449134.
[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, 1995. https://doi.org/10.1007/BF00994018.
[16] X. Gao, W. Bai, Q. Dang, S. Yang, and G. Zhang, "Learnable self-supervised support vector machine based individual selection strategy for multimodal multi-objective optimization," *Inf. Sci. (Ny).*, vol. 690, p. 121553, 2025. https://doi.org/10.1016/j.ins.2024.121553.
[17] I. Rish, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001, vol. 3, no. 22, pp. 41–46.
[18] C. Oddleifson, S. Kilgus, D. A. Klingbeil, A. D. Latham, J. S. Kim, and I. N. Vengurlekar, "Using a naive Bayesian approach to identify academic risk based on multiple sources: A conceptual replication," *J. Sch. Psychol.*, vol. 108, p. 101397, 2025. https://doi.org/10.1016/j.jsp.2024.101397.
[19] T. Sharma, U. Upadhyay, and G. Bagler, "Classification of cuisines from sequentially structured recipes," in *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, 2020, pp. 105–108. https://doi.org/10.1109/ICDEW49219.2020.00008.
[20] L. Hu, M. Jiang, X. Liu, and Z. He, "Significance-based decision tree for interpretable categorical data clustering," *Inf. Sci. (Ny).*, vol. 690, p. 121588, 2025. https://doi.org/10.1016/j.ins.2024.121588.
[21] B. Mallala, A. I. U. Ahmed, S. V Pamidi, M. O. Faruque, and R. Reddy, "Forecasting global sustainable energy from renewable sources using random forest algorithm," *Results Eng.*, vol. 25, p. 103789, 2025. https://doi.org/10.1016/j.rineng.2024.103789.
[22] S. Palaniappan, R. Logeswaran, A. Velayutham, and B. N. Dung, "Predicting short-range weather in tropical regions using random forest classifier," *J. Informatics Web Eng.*, vol. 4, no. 1, pp. 18–28, 2025. https://doi.org/10.33093/jiwe.2025.4.1.2.
[23] Z. Yu, Q. Kanwal, M. Wang, A. Nurdiawati, and S. G. Al-Ghamdi, "Spatiotemporal dynamics and key drivers of carbon emissions in regional construction sectors: Insights from a Random Forest Model," *Clean. Environ. Syst.*, vol. 16, p. 100257, 2025. https://doi.org/10.1016/j.cesys.2025.100257.
[24] W. Tao, Z. Sun, Z. Yang, B. Liang, G. Wang, and S. Xiao, "Transformer fault diagnosis technology based on AdaBoost enhanced transferred convolutional neural network," *Expert Syst. Appl.*, vol. 264, p. 125972, 2025. https://doi.org/10.1016/j.eswa.2024.125972.
[25] C.-Y. Shih, Y.-T. Lin, W. Chen, and J.-C. Huang, "SVM-Adaboost based badminton offensive movement parsing technique," *Signal, Image Video Process.*, vol. 19, no. 4, p. 280, 2025. https://doi.org/10.1007/s11760-025-03865-7.
[26] H. Attou, A. Guezzaz, S. Benkirane, and M. Azrour, "A New Secure Model for Cloud Environments Using RBFNN and AdaBoost," *SN Comput. Sci.*, vol. 6, no. 2, p. 188, 2025. https://doi.org/10.1007/s42979-025-03691-1.
[27] D. Batra *et al.*, "RecipeDB: a resource for exploring recipes," *Database*, vol. 2020, p. baaa077, 2020. https://doi.org/10.1093/database/baaa077.
[28] P. Piplani, P. Gulati, S. Malik, S. Goyal, M. Gurbaxani, and G. Bagler, "FoodPrint: computing carbon footprint of recipes," in *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*, 2022, pp. 95–100. https://doi.org/10.1109/ICDEW55742.2022.00020.
[29] M. Goel *et al.*, "Ratatouille: a tool for novel recipe generation," in *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*, 2022, pp. 107–110. https://doi.org/10.1109/ICDEW55742.2022.00022.
[30] L. V Fausett, *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India, 2006.
[31] D. Meyer, F. Leisch, and K. Hornik, "The support vector machine under test," *Neurocomputing*, vol. 55, no. 1–2, pp. 169–186, 2003. https://doi.org/10.1016/S0925-2312(03)00431-4.

`

[32] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Mach. Learn.*, vol. 29, pp. 103–130, 1997. https://doi.org/10.1023/A:1007413511361.
[33] K. K. Rana, "A survey on decision tree algorithm for classification," *Int. J. Eng. Dev. Res.*, vol. 2, no. 1, pp. 1–5, 2014.
[34] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001. https://doi.org/10.1023/A:1010933404324.
[35] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Stat. Interface*, vol. 2, no. 3, pp. 349–360, 2009.
[36] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances," in *COMPSTAT 1982 5th Symposium held at Toulouse 1982: Part I: Proceedings in Computational Statistics*, 1982, pp. 30–41. https://doi.org/10.1007/978-3-642-51461-6_3.
[37] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.
[38] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. https://doi.org/10.1038/nature14539.
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, 2017. https://doi.org/10.1145/3065386.
[40] M. H. Nguyen and F. de la Torre, "Optimal feature selection for support vector machines," *Pattern Recognit.*, 2010. https://doi.org/10.1016/j.patcog.2009.09.003.
[41] W. H. Chen, J. Y. Shih, and S. Wu, "Comparison of support-vector machines and back propagation neural networks in forecasting the six major Asian stock markets," *Int. J. Electron. Financ.*, 2006. 10.1504/IJEF.2006.008837.
[42] S. R. Sain and V. N. Vapnik, "The Nature of Statistical Learning Theory," *Technometrics*, 1996. 10.2307/1271324.
[43] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," 2006. https://doi.org/10.1145/1143844.1143865.
[44] N. Ketkar and J. Moolayil, *Deep Learning with Python*. 2021.
[45] Y. Tang and R. Salakhutdinov, "Learning stochastic feedforward neural networks," 2013.
[46] C. Robert, "Machine Learning, a Probabilistic Perspectiv,," *CHANCE*, 2014. https://doi.org/10.1080/09332-480.2014.914768.
[47] J. W. Osborne, "Improving your data transformations: Applying the Box-Cox transformation," *Pract. Assessment, Res. Eval.*, 2010.
[48] J. D. M. Rennie, L. Shih, J. Teevan, and D. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," 2003.
[49] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," *AAAI/ICML-98 Work. Learn. Text Categ.*, 1998. 10.1.1.46.1529.
[50] L. E. Raileanu and K. Stoffel, "Theoretical comparison between the Gini Index and Information Gain criteria," *Ann. Math. Artif. Intell.*, 2004. https://doi.org/10.1023/B:AMAI.0000018580.96245.c6.
[51] Y. Goldberg, "A primer on neural network models for natural language processing," *J. Artif. Intell. Res.*, 2016. https://doi.org/10.1613/jair.4992.
[52] T. K. Ho, "Random decision forests," 1995. https://doi.org/10.1109/ICDAR.1995.598994.
[53] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, 2002.
[54] Thomas G. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning*. 2000.
[55] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2003. https://doi.org/10.1007/3-540-36434-x_4.

BIOGRAPHIES OF AUTHORS

Yonathan Ferry Hendrawan is a lecturer in the Informatics Department, Engineering Faculty, Trunojoyo University, Bangkalan, Indonesia. He received his Electrical Engineering Bachelor Degree from Sepuluh Nopember Institute of Technology (ITS), Surabaya in 2004. He received his Information Technology Master's Degree from the University of New South Wales (UNSW), Sydney in 2012. He is currently a PhD candidate at the University of Oklahoma. His research interest is in Information Visualization, Computer Graphics, and Artificial Intelligence. He can be contacted at email: yonathan.hendrawan@trunojoyo.ac.id.

Omkar Chekuri is currently a PhD candidate at University of Oklahoma, where he received M.S in data science and analytics in 2018. His technical interests include information visualization, machine learning, software engineering, direct manipulation user interfaces and data pipeline architectures for information visualizations. He can be contacted at email: omkar.chekuri@ou.edu

`